Analysing MTL Properties using NuSMV Model Checker

Shreya V, Manju Nanda

***Abstract*— Reliability and safety property of any hardware is an important parameter. To achieve this and to improve the performance bounds of the designed system it is important to enhance the efficiency by proper verification techniques. To overcome the problems arising due to the software crisis, formal methods are used. The use of formal methods in aerospace domain is the latest research that is being carried out. Formal verification, a part of formal methods is a mathematical modelling technique used to verify the hardware systems. Technique such as model checking is used to efficiently bridge the gap between design and developed stage of the system with less errors and more efficiency. In this paper, we propose to use NuSMV for verifying the vertical mode functionality of the Mode Transition Logic (MTL). MTL is a very critical functionality in aircraft. It assists the control of trajectories, weather and systems. The NuSMV model checker is used to analyse the functional behaviour of the model. The model is initially designed and developed using Mat-lab/Simulink tool suite. The semantic translation of the MTL model to NuSMV is done by means of specification languages such as CTL and LTL. Test cases generated at the Simulink model level are used as a reference to test the linear and non-linear properties of the MTL vertical model in NuSMV. These test cases are compared with the results obtained using NuSMV analysis. The efficiency is defined by earlier fault detection and improving the software development life cycle of the system.**

Keywords—Model Checking; Autopilot System; Mode Transition Logic; Formal Methods; Formal Verification.

I. INTRODUCTION

Formal methods are techniques used to improve the safety and liveness property in any safety-critical system. To prevent the system failure caused due to errors that are unnoticed during the verification process performed by traditional verification techniques, formal methods are used thereby reducing the percentage of errors by detecting them in the earlier stages of the system development cycle [2]. Formal methods [5] implements the use of formal verification that represents all the models by means of a mathematical logic and replaces all the traditional verification techniques and are mainly applicable and efficient to verify large and complex systems. Formal methods are mainly developed to be used in mission critical and safety critical system where the cost of fault is too high. Safety-critical systems [3] are those system who's malfunctioning or failures may cause catastrophic errors leading to serious death or injuries. These systems have more freedom to expose to danger.

Autopilot is a system which belongs to the category of safety-critical system. It is a system that is mainly used to control the trajectory of the vehicle without the control of a human operator [4]. Autopilots assist the humans in controlling the vehicle, focusing on broader aspects of operation like monitoring the trajectory, weather etc. Modern autopilots use computer software for controlling the aircraft. The software reads the current position and later controls a flight control system to guide the aircraft. In order to develop these systems to an extent that any external or internal conditions do not affect the working of the system, verification of the model should be properly done. The formal verification technique used in this paper is the model checking approach used for verifying complex hardware systems before they are dispatched for fabrication. NuSMV (New Symbolic Model Verifier) [6] is a symbolic model verifier used to analyse the verification status of the model under consideration i.e. the Mode Transition Logic (MTL) of an autopilot system.

Shreya V, M.Tech in Digital Communication, Dept. of TCE, RVCE, Bengaluru, India (shreyavswamy@gmail.com)

Manju Nanda, Principle Scientist
Aerospace Electronics and Systems Division, NAL , Bengaluru, India (manjun@nal.res.in)

II. LITERATURE SURVEY

The discussion of formal methods for safety critical systems by Liu et al., [3] gave an idea about the challenges faced during its implementation process. They mainly concentrated on three problems faced. First of all, there was no well-defined language or principles to describe the mapping from physical environments. Secondly, safety requirements are normally expressed in natural language which is not useful for software developers because it does not provide a firm support for proving whether the system satisfies the required properties. And third problem is that the precise relationship between safety and functionality of the system is not consistently used as a framework for software development for safety critical systems in the whole software life cycle. Also from the conclusion it says, formal methods can be an effective tool in increasing the confidence of the software implementation for safety critical systems.

As pointed out by Bowen in [11] Formal Methods are techniques and tools based on mathematics and formal logic. It requires mathematical expertise but in case of safety-critical system it is highly desirable to obtain benefits from formal methods even in constraints contexts. Use of formal methods in system production delivers enhanced quality as well as correctness i.e. adherences to some requirements.

Yalin Hu [2] pointed out the details of formal verification. It says that, formal verification emerged as an alternative approach for traditional validation techniques such as testing and simulation. Formal verification is an act of proving or disproving the correctness of the algorithms lying within the hardware systems with respect to formal specification or property. Formal verification is a must for safety-critical systems to ensure that the system works properly under the given conditions, if failed to satisfy will cause catastrophic errors in the environment as well.

Formal verification [7] methods such as model checking are used more as industrial application which goes to show the importance and practicality of such methods for real-

978-1-5090-0774-5/16/$31.00 © 2016 IEEE

time embedded systems and System-on Chip (SoC) designs. For the above reasons, we will thus employ a widely popular formal verification method called model checking for the verification of safety-critical systems that are formally modelled.

III. PROPOSED METHODOLOGY

MTL file designed and developed in Matlab/Simulink is simulated and test cases are generated for the same with the help of which the behavioural analysis of the model can be performed. The flowchart for the proposed analysis is as shown in Figure 1. As mentioned, the use of formal methods in the domain of safety engineering has led to the development of latest technologies available for verifying the models to note down the efficiency of the approach in aerospace domain. Henceforth, model checking is a formal verification approach that is been used for analysing the behaviour of these models.

The state transition analysis for variable input conditions can be shown and these results are compared with those obtained by using Matlab/Simulink. The efficiency of the tool for the given model can be evaluated based on this approach.

The specifications are provided to design and verify the model. Once the model is simulated, it generates test cases to find out detailed analysis of the model in terms of various parameters. Using these, state space analysis is done to realise the performance of the model. These specifications are simultaneously given to the model checking tool NuSMV in the form of an input code written with the extension .smv. This code is compiled and simulated to generate traces which determines the truth or falsity of the program. The model is checked for properties using specification languages such as Computation Tree Logic (CTL) and Linear Temporal Logic (LTL) [11]. Once the property check is finished, the results are compared with those obtained from Simulink. If any corrections are present then the model undergoes re-verification and analysis. Else, the efficiency of the model is determined.

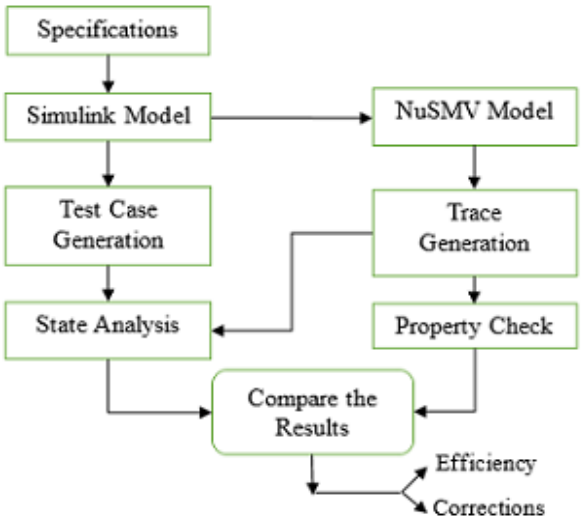

Fig. 1: Design Flowchart

IV. UNIQUENESS OF PROPOSED METHODOLOGY

Mode Transition Logic in autopilot systems gives the entry and exit criteria of various modes according to their performance bounds. Upon certain conditions and requirements it can be changed to other modes. MTL is a discrete event based system consisting of states, events and outputs. Vertical mode [9] mainly implies that it is longitudinal mode which includes Pitch Attitude Hold (PAH) controller. The transition depends on the conditions at the instant of engagement. The vertical modes of autopilot is as shown in Figure 2.

Fig. 2: Vertical Modes in Autopilot [4]

Depending upon the state transition and the conditions, the modes in an autopilot system changes. Matrix indicating the transitions is as shown in Figure 3. Initially when the autopilot system is activated it goes to roll hold i.e. Roll Attitude Mode (RAH) for lateral mode and Pitch Attitude Hold (PAH) mode. The modes are activated manually or automatically depending upon the type of engagement. The first column indicates the state events of the mode. Triggering of the event gives the state change from one value to another. The Figure mainly indicates the total number of transitions from one mode to another.

Sr. No.	EVENT	1	2	3	4	5	6	7	8	9	10	11	12	13
		AP	SPD	VS	ALT	FLC	APPR	VNV	GA	ALTS	FD	SYNC	VD to	DECAP
1	DIS	2	0	0	0	0	0	0	0	0	2	2	0	0
2	PAH	10202	3	4	6	7	8	9	1	2	10202	1	0	1
3	SPD	10203	2	4	6	7	8	9	1	3	10203	1	0	1
4	VS	10204	3	2	6	7	8	9	1	4	10204	1	0	1
5	ALTS	10205	3	4	6	7	8	9	1	2	10205	1	0	1
6	ALT	10206	3	4	2	7	8	9	5	2	10206	1	0	1
7	FLC	10207	3	4	5	2	8	9	1	5	10207	1	0	1
8	GS	10208	3	4	5	7	2	2	1	2	10208	1	0	1
9	VNAV	10209	3	4	6	7	8	9	10	5	10209	1	12	1
10	Go Around	210	2	2	2	2	2	2	1	2	1	2	0	1
11	VPTH	10209	2	4	6	7	8	2	10	5	10209	1	12	1
12	ALTV	10209	2	4	6	7	8	2	10	5	10209	12	6	1

Fig. 3: State Transition Matrix [9]

This analogy is implemented to undergo verification process in the model checking tool NuSMV. NuSMV is a symbolic model verifier with the advantages of having Binary Decision Diagrams (BDD) based and Propositional SATisfiability (SAT) based approach [6] for modelling the program. For the given specifications, input file is analysed to construct the internal representation of the system. NuSMV has the advantage of implementing both synchronous and asynchronous system and also for modelling concurrent systems.

Properties are verified by means of property specification languages such as CTL and LTL that defines the

temporal behaviour of the system. This functional analysis helps to explore the system in a thorough fashion. One more advantage of using model checking over Matlab is that, all the states are analysed simultaneously which is not possible in case of Matlab and the simulation can be performed from any state of interest. It also cross checks whether the execution satisfies the requirement if the transition state satisfies the requirement. Dynamic analysis and verification is performed in case of NuSMV which is advantageous in a concept that it tests and evaluates the model by executing the data in real time thereby analysing the model behaviour. This helps in finding more bugs and faults in the system in the earlier stages of the software life cycle of the product. The block diagram for a model checking approach is shown in Figure 4.

The requirements for any model is specified in terms of a property specification language such as CTL or LTL. Use of LTL is the improvement in the newer versions of NuSMV. Using these languages, the properties can be verified in the model checker. The model requirements are formalised to reduce the complexity of the model and are then developed using specification languages. Similarly, the system undergoes modelling and is then verified with the specifications in the other side and both are given to a model checker. This does the work of verifying whether the properties holds good for the system or not. If it is not satisfied, the errors are identified and corrected by generating a counter example to locate the position of the errors and are simulated again.

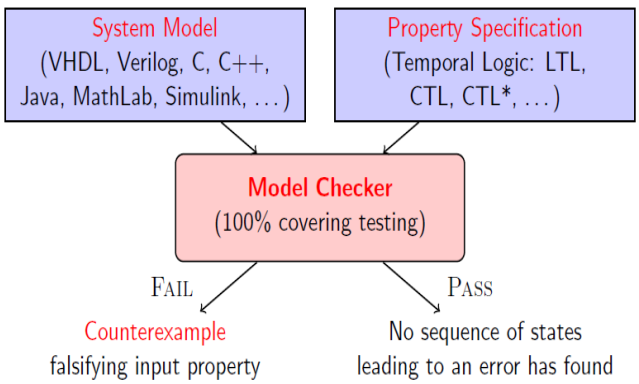

Fig 4: Model Checking Approach [1]

V. RESULTS AND DISCUSSION

The vertical autopilot mode developed in Matlab is analysed and simulated for the generation of test cases. The analysis report is shown in Table 1 where the behavioural outcomes such as test conditions, total analysis time, coverage area, transition conditions, sub-state analysis etc. are all analysed.

The objectives satisfied by the model is generated as a report and is shown in Table 2. Here, the first column represents the type of the model and the second column represents its description that is taking place at that instance. The total time taken for analysing this condition is given in column three and the respective test case for this condition is mentioned in the last column. This basically provides the coverage conditions for the model.

Based on the event indicated by 'e' and the condition indidcated by 'c' the transition condition is defined also stating if the condition is true or false. This exists for various other substates in the model. This description is given in the Table 2(a). In this Table the transition between the substates within the vertical mode is shown. The substate transition also has a condition that if the operation in the previous mode exits only then the state has a change to the next state. In the same way the test cases are generated to verify if the model condition are satisfied accordingly.

The Test Case 5 generated for this model is shown in Table 3. The time taken for the generation of this test case is mentioned. The objectives that are satisfied for a given condition mentioned in the Model item column can be cross verified in the simulink model.

TABLE 1: Analysis for MTL

Analysis Options

Mode:	TestGeneration
Test Suite Optimization:	CombinedObjectives (Nonlinear Extended)
Maximum Testcase Steps:	500time steps
Test Conditions:	UseLocalSettings
Test Objectives:	UseLocalSettings
Model Coverage Objectives:	ConditionDecision
Maximum Analysis Time:	300s
Block Replacement:	off
Parameters Analysis:	off
Randomize data that do not affect the outcome:	off
Save Data:	on
Save Harness:	off
Save Report:	off

TABLE 2: Test and Transition Analysis for Vertical Mode Autopilot

Condition	MTL.MTL.Vm."[e==1&&c2==1]"	Transition: Condition 1, "e==1" T	2	1
Condition	MTL.MTL.Vm."[e==1&&c2==1]"	Transition: Condition 1, "e==1" F	1	1
Condition	MTL.MTL.Vm."[e==1&&c2==1]"	Transition: Condition 2, "c2==1" T	7	1
Condition	MTL.MTL.Vm."[e==1&&c2==1]"	Transition: Condition 2, "c2==1" F	2	1
Decision	MTL.MTL.Vm."[e==1&&c2==1]"	Transition: Transition trigger expression F	1	1
Decision	MTL.MTL.Vm."[e==1&&c2==1]"	Transition: Transition trigger expression T	7	1
Condition	MTL.MTL.Vm."[e==11&&c24==1]"	Transition: Condition 1, "e==11" T	1	1
Condition	MTL.MTL.Vm."[e==11&&c24==1]"	Transition: Condition 1, "e==11" F	1	1
Condition	MTL.MTL.Vm."[e==11&&c24==1]"	Transition: Condition 2, "c24==1" T	6	1
Condition	MTL.MTL.Vm."[e==11&&c24==1]"	Transition: Condition 2, "c24==1" F	1	1
Decision	MTL.MTL.Vm."[e==11&&c24==1]"	Transition: Transition trigger expression F	1	1
Decision	MTL.MTL.Vm."[e==11&&c24==1]"	Transition: Transition trigger expression T	6	1
Condition	MTL.MTL.Vm."[e==1&&c12==1]"	Transition: Condition 1, "e==1" T	10	5
Condition	MTL.MTL.Vm."[e==1&&c12==1]"	Transition: Condition 1, "e==1" F	7	1
Condition	MTL.MTL.Vm."[e==1&&c12==1]"	Transition: Condition 2, "c12==1" T	10	5
Condition	MTL.MTL.Vm."[e==1&&c12==1]"	Transition: Condition 2, "c12==1" F	10	6
Decision	MTL.MTL.Vm."[e==1&&c12==1]"	Transition: Transition trigger expression F	7	1
Decision	MTL.MTL.Vm."[e==1&&c12==1]"	Transition: Transition trigger expression T	10	5
Condition	MTL.MTL.Vm."[e==9&&c23==1]"	Transition: Condition 1, "e==9" T	9	4
Condition	MTL.MTL.Vm."[e==9&&c23==1]"	Transition: Condition 1, "e==9" F	7	1

TABLE 2(a): Test and Transition Analysis for Vertical Mode Autopilot

#	Type	Model Item	Description	Analysis Time (sec)	Test Case
1	Decision	MTL.MTL.Vm	State: Substate executed State "Alt"	6	1
2	Decision	MTL.MTL.Vm	State: Substate executed State "Pah"	5	1
3	Decision	MTL.MTL.Vm	State: Substate executed State "Spd"	14	11
4	Decision	MTL.MTL.Vm	State: Substate executed State "Vm_Dis"	0	1
5	Decision	MTL.MTL.Vm	State: Substate executed State "Vm_Sync"	6	1
6	Decision	MTL.MTL.Vm	State: Substate executed State "Vs"	13	10
7	Decision	MTL.MTL.Vm	State: Substate exited when parent exits State "Alt"	117	92
8	Decision	MTL.MTL.Vm	State: Substate exited when parent exits State "Pah"	5	1
9	Decision	MTL.MTL.Vm	State: Substate exited when parent exits State "Spd"	14	11
10	Decision	MTL.MTL.Vm	State: Substate exited when parent exits State "Vm_Dis"	2	1
11	Decision	MTL.MTL.Vm	State: Substate exited when parent exits State "Vm_Sync"	74	62
12	Decision	MTL.MTL.Vm	State: Substate exited when parent exits State "Vs"	79	65

TABLE 3: Test Cases number 5 for the given condition

Step	Time	Model Item	Objectives
79	15.6	MTL.MTL.Vm."[e==1&&c12==1]"	Transition: Transition trigger expression T
		MTL.MTL.Vm."[e==1&&c12==1]"	Transition: Condition 1, "e==1" T
		MTL.MTL.Vm."[e==1&&c12==1]"	Transition: Condition 2, "c12==1" T
		MTL.MTL.Ap."[e==1&&c12==1]"	Transition: Transition trigger expression T
		MTL.MTL.Ap."[e==1&&c12==1]"	Transition: Condition 2, "c12==1" T

VI. CONCLUSION

In this paper, details about the vertical mode in MTL of the autopilot system is discussed. The state change depending upon the conditions provided by the specification The model analysis is done in Matlab and the model conditions are verified by generating the test cases for various events, states and transitions happening in the MTL model. This model is implemented to check the functional and temporal behaviour in model checking tool NuSMV. This technique is suitable for large models having a complex design with more number of transition states. Flexible and robust technique for dynamic verification of the model as compared to other model checkers. The property check is improved by using the specification languages such as LTL and CTL. If the specification satisfies the given property or not is verified using this approach.

ACKNOWLEDGMENT

I thank my guide for enlightening me with the latest research topics such as use of formal methods in aerospace engineering for simulation and verification of hardware modules and also the department of CSIR-NAL for providing an opportunity to gain knowledge and experience in this domain, and anonymous referees for their valuable comments, acknowledgments and suggestions.

REFERENCES

[1] Christer Baier, and Joost-Pieter Kateon, Book on "Principles of Model Checking," Representation in Mind Series, 2008, MIT Press, ACM Digital Library.

[2] Jonathen Bowen, "Safety-Critical Systems, Formal Methods and Standards," Software Engineering Journal, IET Publications, Aug 2002, pp. 189-209.

[3] Shaoying Lin, Victoria Stavridon, and Bruno Dutestre, "The Practice of Formal Methods in Safety-Critical System," Journals of Systems and Software, Elsevier Publications, Jan 1995, pp. 77-87.

[4] Princy Randhawa, Atit Mishra, Yogananada Jeppu, C.G. Nayak, and Nagaraj Murthy, "Mode Transition Logic of a Vertical Autopilot for Commercial Aircraft," *in the Proceedings of IX Control Instrumentation System Conference,* Manipal Institute of Technology, Manipal, Nov 2012.

[5] S.Ray, "An Overview of Formal Methods, Rigorous Software Development: An Introduction to Program Verification," Book, Spinger Publications, ISBN: 978-0-85729-081-2, 2011, pp. 15-44.

[6] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella, "NuSMV 2: An Open Source Tool for Symbolic Model Checking," in *Proceeding of 14th International Conference on Computer Aided Verification, CAV'02,* London, UK, pp. 359-364, 2002.

[7] Anne E. Haxthansen, "An Introduction to Formal Methods for the Development of Safety-Critical Applications," Journal Paper, Technical University of Denmark Journal, Aug 2010.

[8] J. C. Bicarregui, J. S. Fitzgerald, P. G. Larsen, and J. C. P. Woodcock, "Industrial Practice in Formal Methods: A Review," in *Proceedings of 10th Annual Springer Conference on Computer Science and Technology, ASCCST,* New York, vol. 5850, pp. 810-813, Dec. 2009.

[9] Aparna S Nair, Yogananda Jeppu, and C.G Nayak, "Logic for Mode Transition of Autopilots in Lateral Direction for Commercial Aircrafts," Bonfring International Journal of Man Machine Interface, ISSN: 2277-5064, Mar 2013, Vol. 3, pp. 01-07.

[10] Yalin Hu, "Exploring Formal Verification Methodology for FPGA-based Digital Systems," Sandia National Laboratories, New Mexico, California, Sep. 2012.

[11] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "NuSMV: A New Symbolic Model Verifier," in *Proceedings of 11th International Conference on Computer Aided Verification, CAV'99,* Trento, Italy, vol. 1633, pp. 495-499, Jul. 6-10, 1999.

[12] Alexey Voronov and Knut A kesson, "Verification of Process Operation Using Model Checking," in *Proceedings of 5th Annual IEEE Conference on Automated Science and Engineering*, ASC, Bengaluru, India, Aug. 22-25, 2009.